Proportional control for 1st-order system, theoretical analysis (symbolic toolbox) G(s)=2/(2*s+1)

Objective: understand the behaviour of proportional control (simulations in companion videos) on a 1st-order system.

Presentations in video:

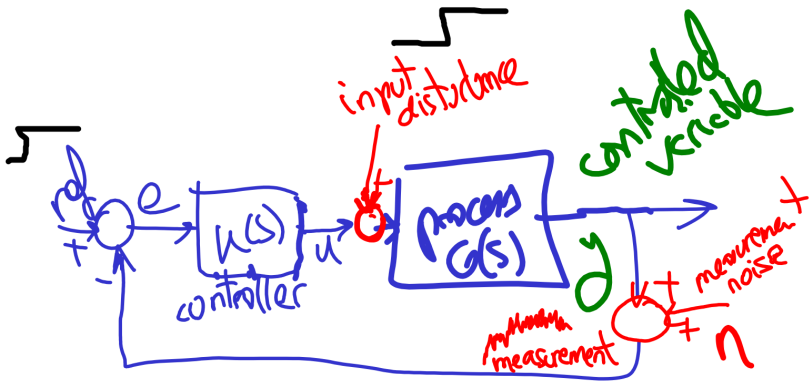
https://personales.upv.es/asala/YT/V/Pteo1EN.html (setpoint tracking)

https://personales.upv.es/asala/YT/V/Pteo2EN.html (input disturbance/noise)

Remark: as controller gain will be a "decision variable", we'll keep it "symbolic" and use the "symbolic Math toolbox" for "computer algebra" operations. For a GIVEN value of controller gain, there is a much friendlier interface in the "control systems toolbox" (and even for control design).

Closed-loop transfer functions

In the closed-loop below:



The equations are:

```
syms G K r u y n du e
LoopEquations={u == K*e,  y == G*(u+du), e == r-(y+n)};
```

Let us solve them:

```
sol=solve(LoopEquations, [u y e])
```

```
sol = struct with fields:
    u: -(K*(n - r + G*du))/(G*K + 1)
    y: (G*(du - K*n + K*r))/(G*K + 1)
    e: -(n - r + G*du)/(G*K + 1)
```

We will be interested in output (**controlled** variable) trajectories:

```
collect(sol.y,[r du n])
```

```
ans =
```

$$\frac{GK}{GK+1}\,r+\frac{G}{GK+1}\,\mathrm{du}+\left(-\frac{GK}{GK+1}\right)n$$

Or, well, error will be $r-measurement$, but, anyway, we can display it:

```
collect(sol.e,[r du n])
```

```
ans =
```

$$\frac{r}{GK+1}+\left(-\frac{G}{GK+1}\right)\mathrm{du}+\left(-\frac{1}{GK+1}\right)n$$

And, later on, in **manipulated** variable trajectories:

```
collect(sol.u,[r du n])
```

```
ans =
```

$$\frac{K}{GK+1}\,r+\left(-\frac{GK}{GK+1}\right)\mathrm{du}+\left(-\frac{K}{GK+1}\right)n$$

Example

```
clear %we discard the above code
syms s %Laplace Variable
syms K_c real %proportional control constant
G(s)=2/(2*s+1);
DCgainG=G(0)
```

```
DCgainG = 2
```

```
K=K_c; %Proportional Control (we might try other expressions in the
future)
```

Setpoint change response

```
CLref(s)=collect(simplify(G*K/(1+G*K)),s)
```

```
CLref(s) =
```

$$\frac{2\,K_c}{2\,s+2\,K_c+1}$$

Dynamics:

```
[N,D]=numden(CLref);
solve(D==0,s) %closed-loop poles
```

```
ans =
```

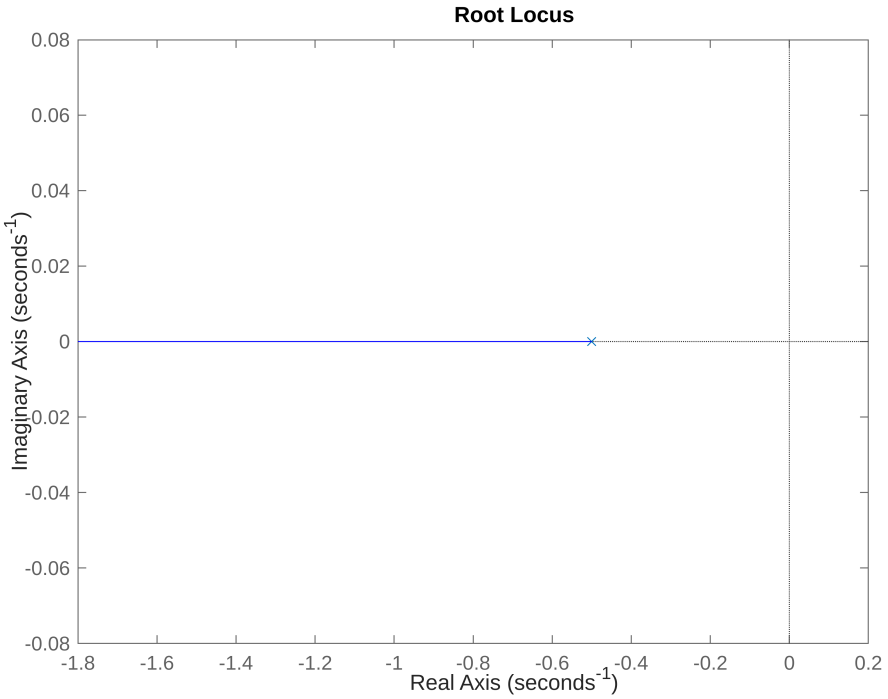
$$-K_c-\frac{1}{2}$$

Then, closed loop pole is $s=-(1/2+K_c)$, settling time (95%) is

$3\tau=3\cdot\frac{1}{|pole|}=3\cdot\frac{1}{0.5+K_c}$. Stable if $K_c>-1/2$... and, come on, if you are not an idiot, for sure you will set $K_c>0$ (push up if output is below setpoint), so "P" controller is STABLE, and the LARGER K_c is, the FASTER the transient is.

To plot the closed-loop poles as a function of K_c we may use the "root locus" command from Control systems Toolbox, because you will likely be familiar with the graph below:

```
rlocus(tf(2,[2 1]))
```



Steady-state error: DC gain of CLref is

```
DCGainREF=CLref(0) %subs(CLref,s,0)
```

```
DCGainREF =
```

$$\frac{2K_c}{2K_c+1}$$

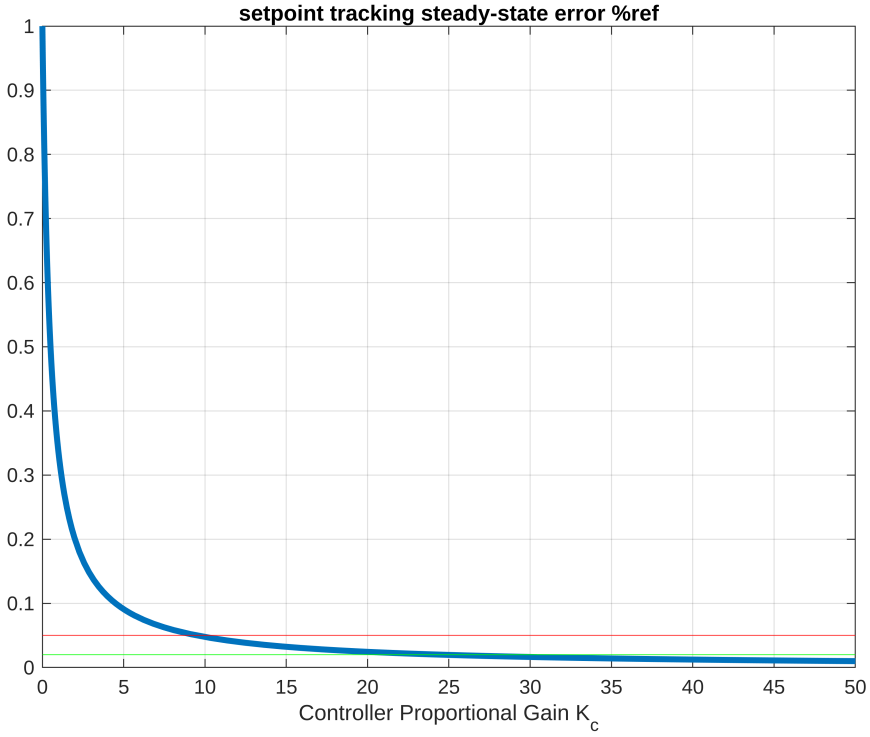
So, final value will be $DCGainREF \cdot r_{finalvalue}$ and steady-state error will be $(1 - DCGainREF) \cdot r_{finalvalue}$, then:

```
DCGainErrREF=simplify(1-DCGainREF)
```

```
DCGainErrREF =
```

$$\frac{1}{2K_c+1}$$

```
fplot(DCGainErrREF,[0 50],LineWidth=3), grid on, title("setpoint
tracking steady-state error %ref")
xlabel("Controller Proportional Gain K_c")
yline(0.05,'r')
yline(0.02,'g')
ylim([0 1])
```



Input Disturbance Response

```
CLdu(s)=collect(simplify(G/(1+G*K)),s)
```

```
CLdu(s) =
```

$$\frac{2}{2s+2K_c+1}$$

Dynamics: closed loop pole identical to setpoint TF.

Steady-steady error:

```
DCGain_Err_du=CLdu(0)
```

```
DCGain_Err_du =
```

$$\frac{2}{2\,K_c+1}$$

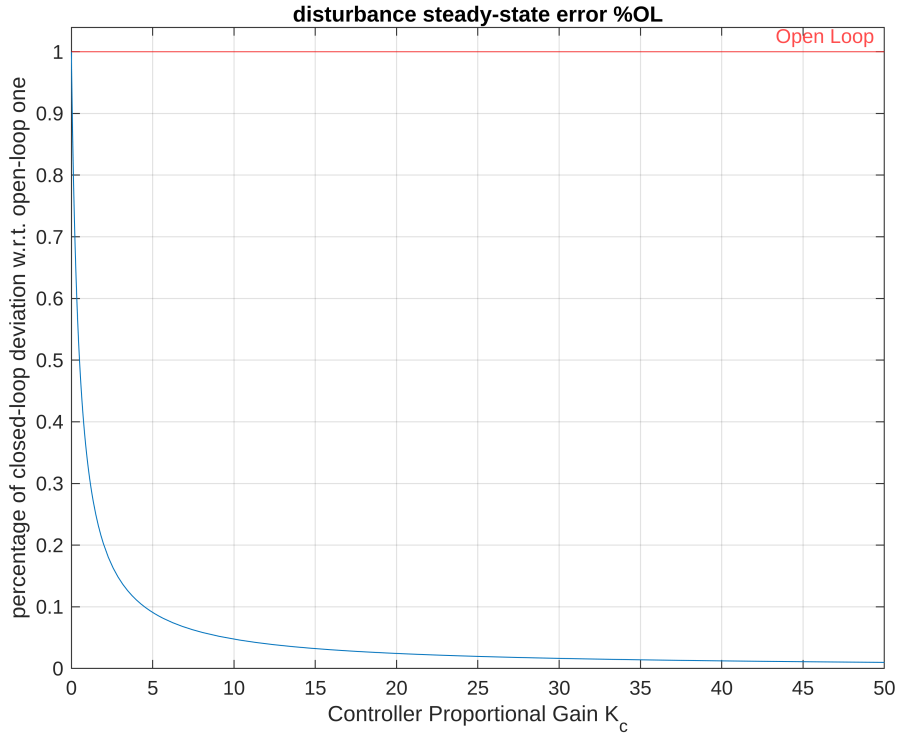
The interesting "adimensional" quantity is the ratio between "no-control open loop" and "closed-loop" error value:

```
RatioCL2OL=DCGain_Err_du/DCgainG
```

```
RatioCL2OL =
```

$$\frac{1}{2\,K_c+1}$$

```
fplot(RatioCL2OL,[0 50]), grid on, title("disturbance steady-state
error %OL")
xlabel("Controller Proportional Gain K_c")
yline(1,'r',label='Open Loop')
ylim([0 1.04]), ylabel("percentage of closed-loop deviation w.r.t.
open-loop one")
```



Measurement Noise response

In this case we are usually interested in high-frequency noise amplification to the actuator (manipulated variable) command:

```
CL_dy_2_u(s)=collect(simplify(K/(1+G*K)),s)
```

```
CL_dy_2_u(s) =
```

$$\frac{(2\,K_c)\,s + K_c}{2\,s + 2\,K_c + 1}$$

```
syms w real
FreqResponse=simplify( abs( CL_dy_2_u(1j*w) ), 100)
```

```
FreqResponse =
```

$$\frac{|K_c|\,\sqrt{4\,|w|^2 + 1}}{\sqrt{(2\,K_c + 1)^2 + 4\,w^2}}$$

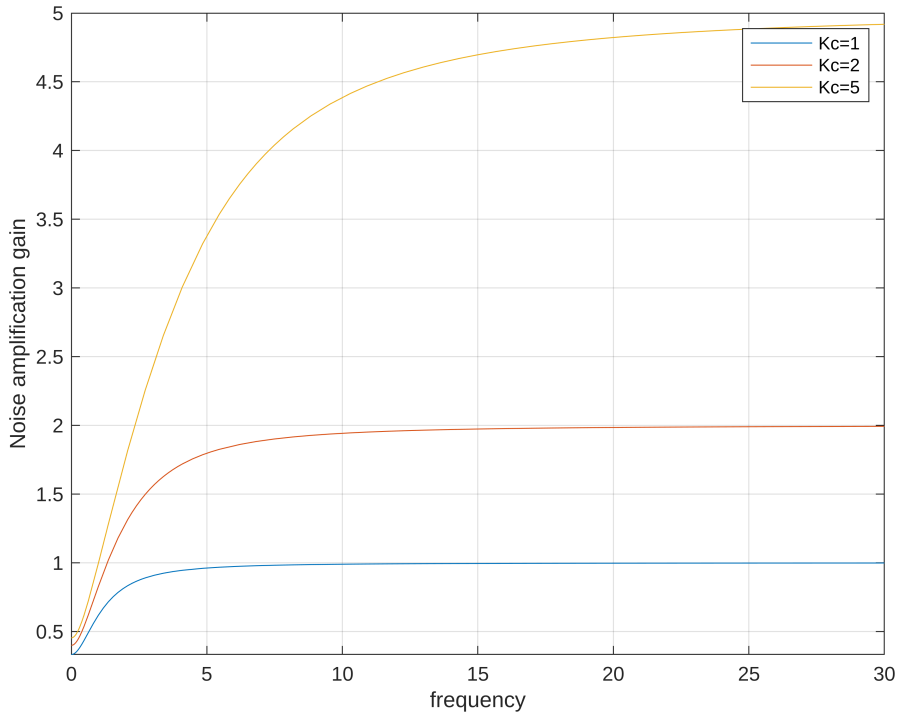
```
limit(FreqResponse,w,inf) %high-frequency amplification
```

```
ans =
```

$|K_c|$

Let us try several values of K_c:

```
f1=subs(FreqResponse,K_c,1);
f2=subs(FreqResponse,K_c,2);
f5=subs(FreqResponse,K_c,5);
fplot([f1 f2 f5],[0 30]), grid on, xlabel("frequency"),
ylabel("Noise amplification gain")
legend("Kc=1","Kc=2","Kc=5")
```



Remark: this is a NON logarithmic plot; usually you will see the "Bode" diagram, which has logarithmic scales both in "frequency" and "amplitude gain" axes.